IS GRAPHQL REALLY HELPFUL IN CONTRAST TO REST API'S?

Daniel Danielecki

WHAT IS A GRAPHQL?

Query language for API's, **alternative** (not replacement) for REST.

GRAPHQL VS REST

GRAPHQL VS REST

Feature	REST	GraphQL	Similar/Differen
HTTP	Yes	Yes	Similar
API endp.	Yes	Yes	Similar
JSON	Yes	Yes	Similar
Request	Rigid	Flexible	Different
Call API	Endpoint	Query	Different
Response	Server	Client	Different
Size resp.	Server	Client	Different

WHY GRAPHQL?

FRONTEND?

WHY GRAPHQL?

Query only what you need.

BACKEND?

WHY GRAPHQL?

- No Over/Under-Fetching
- Non-null
- Performance*
- Single Endpoint
- Strong Typing
- Subscriptions
- Versioning
- ...

GRAPHQL?

(optional)

GRAPHQL BASICS

(optional)

GRAPHQL BASICS: SCHEMA

(optional)

```
type Query { ... }
type Mutation { ... }
type Subscription { ... }
input CreateUserInput { ... }
```

(optional)

GRAPHQL BASICS: OBJECT TYPES

(optional)

GRAPHQL BASICS: OBJECT TYPES

```
type User {
  id: ID!
  name: String!
  email: String!
  age: Int
  posts: [Post!]!
}
type Post {
  id: ID!
  title: String!
  body: String!
  author: User!
}
```

(optional)

GRAPHQL BASICS: QUERY

(optional)

GRAPHQL BASICS: QUERY

```
type Query {
  users(query: String): [User!]!
  posts(query: String): [Post!]!
  dummy: User!
  post: Post!
}
```

(optional)

GRAPHQL BASICS: QUERY

```
query {
  users {
    id
  }
}
```

(optional)

GRAPHQL QUERY == "GET"

(optional)

GRAPHQL MUTATIONS

(optional)

GRAPHQL MUTATIONS

```
type Mutation {
  createUser(data: CreateUserInput!): User!
  deleteUser(id: ID!): User!
  updateUser(id: ID!, data: UpdateUserInput!): User!
}

input CreateUserInput {
  name: String!
  email: String!
  age: Int
}
```

(optional)

GRAPHQL MUTATIONS

```
mutation {
  createUser(data: { name: "Daniel", email: "foo@example.com",
    id
    email
    age
  }
}
```

(optional)

GRAPHQL MUTATION

==

"POST", "PUT", "PATCH", "DELETE"

(CRUD)

(optional)

GRAPHQL SUBSCRIPTIONS

(optional)

GRAPHQL SUBSCRIPTIONS

```
type Subscription {
  count: Int!
}
```

(optional)

GRAPHQL SUBSCRIPTIONS

```
subscription {
  count
}
```

(optional)

GRAPHQL SUBSCRIPTION

==

WEBSOCKETS

(optional)

DEMO?

Simplest example of GraphQL query.

BEYOND THE BASICS

LEARNING CURVE

LEARNING CURVE

GraphQL takes **more time** to learn.

CACHING

CACHING

GraphQL's caching **is more complex**.

CACHING: REST (EXPRESS.JS)

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/api/resource', (req, res) => {
    const resource = {
        id: 1,
        name: "Resource",
        value: "Some data",
        lastModified: new Date().toISOString()
    };

    // Set Cache-Control header
    res.setHeader('Cache-Control', 'public, max-age=3600');
```

CACHING: GRAPHQL (APOLLO SERVER)

```
const { ApolloServer, gql } = require('apollo-server');
const { ApolloServerPluginCacheControl } = require('apollo-ser

// Schema definition
const typeDefs = gql`
  type Query {
    resource(id: ID!): Resource
  }

  type Resource {
    id: ID!
    name: String
    value: String
  }
```

CACHING: GRAPHQL (APOLLO CLIENT)

```
import { ApolloClient, InMemoryCache, gql } from '@apollo/clie

// Initialize Apollo Client
const client = new ApolloClient({
  uri: 'https://example.com/graphql',
  cache: new InMemoryCache()
});

// Query
const GET_RESOURCE = gql`
  query GetResource($id: ID!) {
    resource(id: $id) {
      id
      name
```

ERROR HANDLING

ERROR HANDLING

Beyond 200 when an error occured.

ERROR HANDLING

REST	GraphQL
Status codes to indicate nature of errors	Error classes
Error messages within response body	Error messages within resolvers

ERROR HANDLING: REST

```
try {
  const resource = await Resource.findByPk(id);
  if (resource) {
    res.json(resource);
  } else {
    res.status(404).json({ error: 'Resource not found' });
  }
} catch (error) {
  res.status(500).json({ error: error.message });
}
```

ERROR HANDLING: GRAPHQL

```
try {
  const resource = await Resource.findByPk(id);
  if (resource) {
    return resource;
  } else {
    throw new UserInputError('Resource not found');
  }
} catch (error) {
  throw new Error('Failed to fetch resource');
}
```

DEMO

Let's see an error.

SOME ERRORS
!==
200

- Bad Request: 400
- Unauthorized: 401
- Forbidden: 403
- Internal Server Error: 500

GLOBAL OBJECT IDENTIFICATION

GLOBAL OBJECT IDENTIFICATION

Let my frontend fetch a single ID.

GLOBAL OBJECT IDENTIFICATION: GRAPHQL

```
query {
  user(id: "1") {
    id
    name
    email
  }
}
```

GLOBAL OBJECT IDENTIFICATION: REST

```
GET /resources/:id
```

PAGINATION

PAGINATION

Let my frontend fetch a page.

PAGINATION: GRAPHQL

```
query {
  items(limit: 10, offset: 20) {
    total
    limit
    offset
    items {
      id
      name
    }
  }
}
```

PAGINATION: REST

```
GET /api/items?limit=10&offset=20
```

SORTING

SORTING

Let my frontend sort the results.

SORTING: GRAPHQL

```
query {
  items(sort: { field: "name", order: DESC }) {
    id
    name
  }
}
```

SORTING: REST

```
GET /api/items?sortBy=name&order=desc
```

STACKOVERFLOW

STACKOVERFLOW

What are Developers asking for?

Home
Questions
Staging Ground
Tags
Saves
Users
Companies
LABS
Jobs
Discussions
COLLECTIVES
Communities for your favorite technologies.
Explore all Collectives
TEAMS
Ask questions, find answers and collaborate at work with Stack Overflow for Teams.
Explore Teams
Create a free Team
Looking for your Teams?

Questions tagged [graphql]

Ask Question

GraphQL is an API technology designed to describe the complex, nested data dependencies of modern web applications. It is often considered an alternative to SOAP or REST

Watch tag | Ignore tag

Learn more... Improve tag info Top users Synonyms

21,127 questions

Newest Active Bountied Unanswered More Filter

368 votes
✓ 7 answers
344k views

How to query all the GraphQL type fields without writing a long query?

Assume you have a GraphQL type and it includes many fields. How to query all the fields without writing down a long query that includes the names of all the fields? For example, If I have these ...

php laravel graphql graphql-php

BlackSigma 3,785 asked Dec 10, 2015 at 10:50

276 votes
✓ 12 answers
294k views

Nodemon Error: "System limit for number of file watchers reached"

I'm learning GraphQL and am using prisma-binding for GraphQL operations. I'm facing this node-mon error while I'm starting my Node.js server and its giving me the path of schema file which is...

node.js graphql nodemon

Rehan Sattar 3,825 asked Dec 26, 2018 at 9:50

275 votes
✓ 9 answers
74k views

When and How to use GraphQL with microservice architecture

I'm trying to understand where GraphQL is most suitable to use within a microservice architecture. There is some debate about having only 1 GraphQL schema that works as API Gateway proxying...

architecture microservices graphql

Fabrizio Fenoglio 5,907 asked Jun 28, 2016 at 9:04

216 votes
✓ 4 answers
287k views

How to properly make mock throw an error in Jest?

I'm testing my GraphQL api using Jest. I'm using a separate test suit for each query/mutation I have 2 tests (each one in a separate test suit) where I mock one function (namely, Meteor's call-...

javascript meteor graphql jestjs

Le garcon 7,817 asked Apr 14, 2018 at 19:18

187 votes
✓ 2 answers
73k views

In GraphQL what's the meaning of "edges" and "node"?

I am consuming a GraphQL endpoint and I get results that contain edges and node tags. I am supplying a clean JSON structure for my query, so this doesn't make sense to me. It seems as if the...

json graphql

Ska 6,848 asked Mar 6, 2017 at 10:15

183 votes
What's the point of input type in GraphQL?

The Overflow Blog
Introducing Staging Ground: The private space to get feedback on questions...
Featured on Meta
The 2024 Developer Survey Is Live
The return of Staging Ground to Stack Overflow
The [tax] tag is being burninated
Policy: Generative AI (e.g., ChatGPT) is banned
Hot Meta Posts
4 What should I do when a question in Staging Ground needs migrating?

Custom Filters

Create a custom filter

Watched Tags edit

angular bdd css html javascript node.js single-page-application tdd

Ignored Tags

Add an ignored tag

Home
Questions
Staging Ground
Tags
Saves
Users
Companies
LABS
Jobs
Discussions
COLLECTIVES
Communities for your favorite technologies.
Explore all Collectives
TEAMS
Ask questions, find answers and collaborate at work with Stack Overflow for Teams.
Explore Teams
Create a free Team
Looking for your Teams?

How to query all the GraphQL type fields without writing a long query?

Ask Question

Asked 8 years, 6 months ago Modified 2 days ago Viewed 344k times Part of PHP Collective

368

Assume you have a GraphQL type and it includes many fields. How to query all the fields without writing down a long query that includes the names of all the fields?

For example, If I have these fields :

```
public function fields()
    {
        return [
            'id' => [
                'type' => Type::nonNull(Type::string()),
                'description' => 'The id of the user'
            ],
            'username' => [
                'type' => Type::string(),
                'description' => 'The email of user'
            ],
             'count' => [
                'type' => Type::int(),
                'description' => 'login count for the user'
            ]

        ];
    }
```

To query all the fields usually the query is something like this:

```
FetchUsers{users(id:"2"){id,username,count}}
```

But I want a way to have the same results without writing all the fields, something like this:

```
FetchUsers{users(id:"2"){*}}
//or
FetchUsers{users(id:"2")}
```

Is there a way to do this in GraphQL ??

I'm using *Folkloreatelier/laravel-graphql* library.

PHP php laravel graphql graphql-php

PHP Collective Join the discussion

This question is in a collective: a subcommunity defined by tags with relevant content and experts.

The Overflow Blog

Introducing Staging Ground: The private space to get feedback on questions...

Featured on Meta

The 2024 Developer Survey Is Live

The return of Staging Ground to Stack Overflow

The [tax] tag is being burninated

Policy: Generative AI (e.g., ChatGPT) is banned

Hot Meta Posts

9 Are "Is my understanding correct" questions appropriate?

355

Unfortunately what you'd like to do is not possible. GraphQL requires you to be explicit about specifying which fields you would like returned from your query.

Share Edit Follow Flag

answered Dec 11, 2015 at 15:03

Peter Horne

6,720 ●7 ●39 ●50

17 Ok, and if I request some object of an unknown form from backend which I'm supposed to proxy or send back? – meandre Nov 30, 2016 at 6:25

57 @meandre, the whole idea of graphql is that there is no such thing as an "unkown form". – s.meijer Dec 28, 2016 at 22:16

WON'T WORK

```
query {
  *
}
```

WON'T WORK

```
query {
  users {
    *
  }
}
```

INTROSPECTION

Introspection

It's often useful to ask a GraphQL schema for information about what queries it supports. GraphQL allows us to do so using the introspection system!

For our Star Wars example, the file starWarsIntrospection-test.ts contains a number of queries demonstrating the introspection system, and is a test file that can be run to exercise the reference implementation's introspection system.

We designed the type system, so we know what types are available, but if we didn't, we can ask GraphQL, by querying the `__schema` field, always available on the root type of a Query. Let's do so now, and ask what types are available.

```
{
  __schema {
    types {
      name
    }
  }
}
```

```
        },
        {
          "name": "__DirectiveLocation"
        }
      ]
    }
  }
}
```

Wow, that's a lot of types! What are they? Let's group them:

- `Query`, `Character`, `Human`, `Episode`, `Droid` - These are the ones that we defined in our type system.
- `String`, `Boolean` - These are built-in scalars that the type system provided.
- `__Schema`, `__Type`, `__TypeKind`, `__Field`, `__InputValue`, `__EnumValue`, `__Directive` - These all are preceded with a double underscore, indicating that they are part of the introspection system.

Now, let's try and figure out a good place to start exploring what queries are available. When we designed our type system, we specified what type all queries would start at; let's ask the introspection system about that!

```
{
  __schema {
    queryType {
      name
```

```
"data": {
  "__schema": {
    "queryType": {
      "name": "Query"
```

DEMO

Introspection in practice.

```
const server = new ApolloServer({
  typeDefs,
  resolvers,
  introspection: process.env.NODE_ENV !== 'production'
});
```

GRAPHQL VOYAGER

CHANGE SCHEMA

Type List

Search Schema...

Root root

No Description

Film

A single film.

Person

An individual person or character within the Star Wars universe.

Planet

A large mass, planet or planetoid in the Star Wars Universe, at the time of 0 ABY.

Species

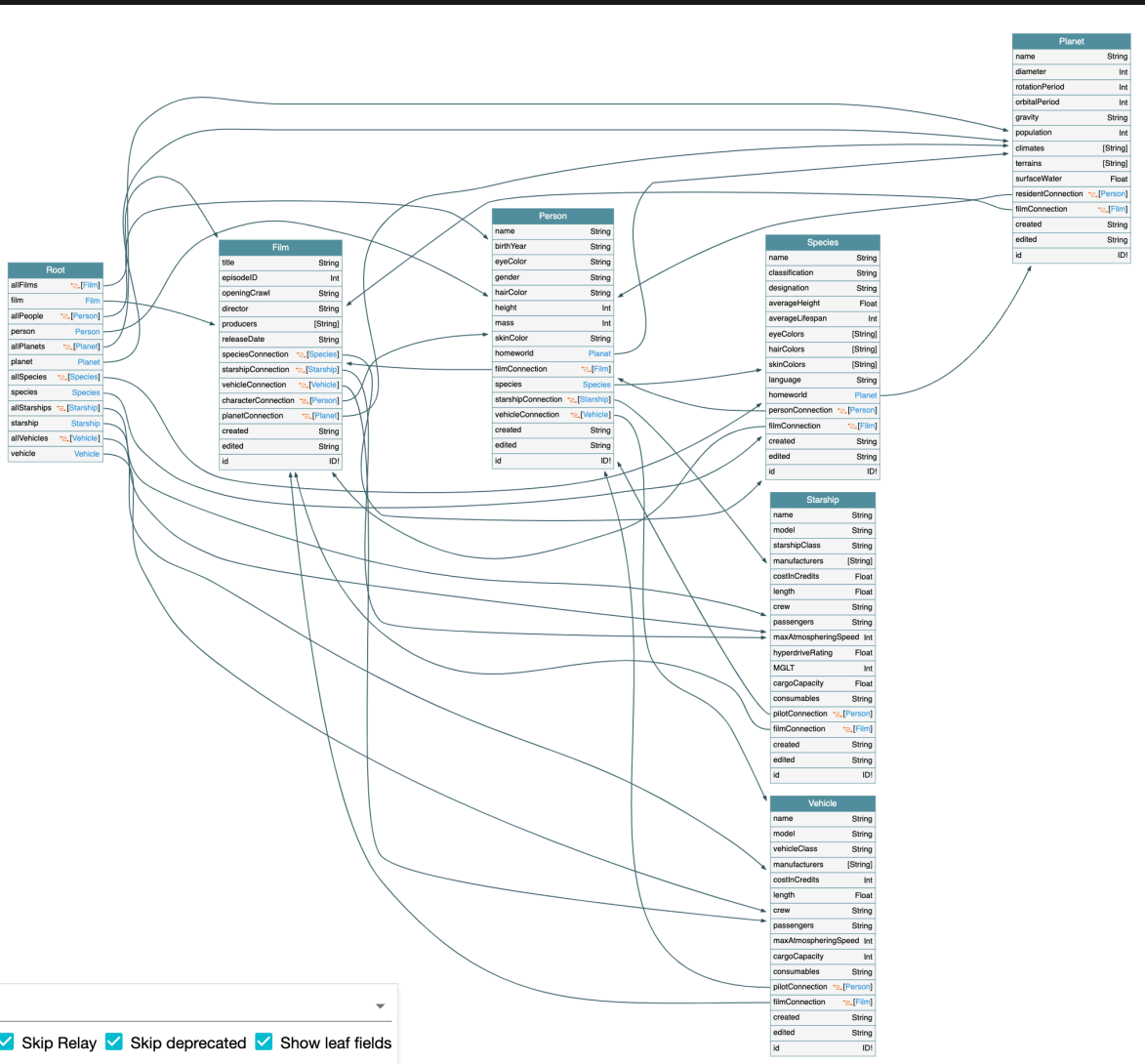
A type of person or character within the Star Wars Universe.

Starship

A single transport craft that has hyperdrive capability.

Vehicle

A single transport craft that does not have hyperdrive capability



Root

Sort by Alphabet | Skip Relay | Skip deprecated | Show leaf fields

RESET

Powered by GraphQL Voyager

WSTG - Latest

Testing GraphQL

ID
WSTG-APIT-01

Summary

GraphQL has become very popular in modern APIs. It provides simplicity and nested objects, which facilitate faster development. While every technology has advantages, it can also expose the application to new attack surfaces. The purpose of this scenario is to provide some common misconfigurations and attack vectors on applications that utilize GraphQL. Some vectors are unique to GraphQL (e.g. Introspection Query) and some are generic to APIs (e.g. SQL injection).

Examples in this section will be based on a vulnerable GraphQL application poc-graphql, which is run in a docker container that maps `localhost:8080/GraphQL` as the vulnerable GraphQL node.

Test Objectives

- Assess that a secure and production-ready configuration is deployed.
- Validate all input fields against generic attacks.
- Ensure that proper access controls are applied.

How to Test

Testing GraphQL nodes is not very different than testing other API technologies. Consider the following steps:

Introspection Queries

Introspection queries are the method by which GraphQL lets you ask what queries are supported, which data types are

WSTG Contents

Summary

Share article

May 7, 2021

Why You Should Disable GraphQL Introspection In Production – GraphQL Security

Khalil Stemmler

CIRCULAR QUERIES

DEMO

Digging with nesting.

STANDARDIZATION VS FRAGMENTATION

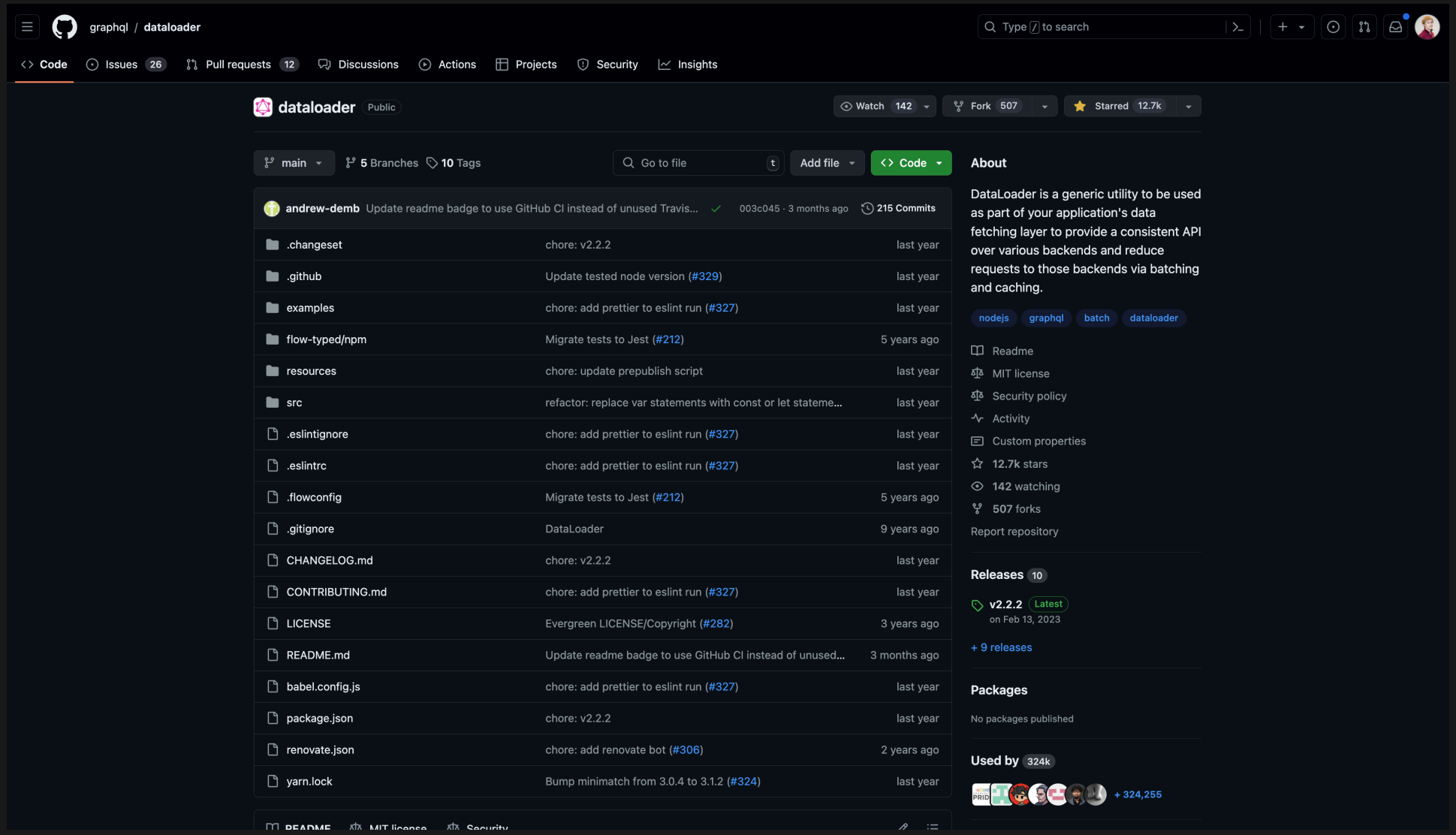
STANDARDIZATION: REST

HTTP methods (GET, POST, PUT, DELETE), status codes (200, 404, 500).

FRAGMENTATION: GRAPHQL

Apollo Server, GraphQL Yoga, Envelop, Express-GraphQL (deprecated), Apollo Client, Relay, urql.

PROJECTS


graphql / dataloader
Type / to search
Code
Issues 26
Pull requests 12
Discussions
Actions
Projects
Security
Insights
dataloader
Public
Watch 142
Fork 507
Starred 12.7k
main
5 Branches
10 Tags
Go to file
t
Add file
Code
andrew-demb Update readme badge to use GitHub CI instead of unused Travis...
003c045 · 3 months ago
215 Commits
.changeset chore: v2.2.2 last year
.github Update tested node version (#329) last year
examples chore: add prettier to eslint run (#327) last year
flow-typed/npm Migrate tests to Jest (#212) 5 years ago
resources chore: update prepublish script last year
src refactor: replace var statements with const or let stateme... last year
.eslintignore chore: add prettier to eslint run (#327) last year
.eslintrc chore: add prettier to eslint run (#327) last year
.flowconfig Migrate tests to Jest (#212) 5 years ago
.gitignore DataLoader 9 years ago
CHANGELOG.md chore: v2.2.2 last year
CONTRIBUTING.md chore: add prettier to eslint run (#327) last year
LICENSE Evergreen LICENSE/Copyright (#282) 3 years ago
README.md Update readme badge to use GitHub CI instead of unused... 3 months ago
babel.config.js chore: add prettier to eslint run (#327) last year
package.json chore: v2.2.2 last year
renovate.json chore: add renovate bot (#306) 2 years ago
yarn.lock Bump minimatch from 3.0.4 to 3.1.2 (#324) last year
README
MIT license
Security
About
DataLoader is a generic utility to be used as part of your application's data fetching layer to provide a consistent API over various backends and reduce requests to those backends via batching and caching.
nodejs
graphql
batch
dataloader
Readme
MIT license
Security policy
Activity
Custom properties
12.7k stars
142 watching
507 forks
Report repository
Releases 10
v2.2.2 Latest
on Feb 13, 2023
+ 9 releases
Packages
No packages published
Used by 324k
+ 324,255

Releases Tags

Find a release

Aug 9, 2023

IvanGoncharov

v2.0.0

1e51d57

Compare

v2.0.0 Latest

v2.0.0 (2023-08-09)

Made in Ukraine 🇺🇦

You can show your support by helping my country to fight this war:
https://opencollective.com/graphql-voyager
All collected money after expenses goes to Ukrainian paramedics deployed alongside troops.

Breaking Change 💥

- #317 Remove 'init' and 'GraphQLVoyager' exports (@IvanGoncharov)
- #318 Remove support for passing function as introspection (@IvanGoncharov)
- #319 Delete `voyager.min.js` from the dist (@IvanGoncharov)

New Feature 🚀

- #323 Drop `react-dom` from peerDependencies (@IvanGoncharov)
- #340 Switch to accepting schema as introspection (@IvanGoncharov)
- #348 Add cache for SVG generate by graphviz (@IvanGoncharov)

Bug Fix 🐞

- #325 Track resize of viewport instead of entire page (@IvanGoncharov)

graphql-kit / graphql-voyager

Type / to search

Code | Issues 87 | Pull requests 21 | Discussions | Actions | Projects | Security | Insights

Voyager fails to load (Unexpected reserved word at voyager:42:29: "await") #403

New issue

Open craigdrayton opened this issue 2 weeks ago · 2 comments

craigdrayton commented 2 weeks ago

I'm trying to use Voyager with a GraphQL Yoga server running Express:

```
import express from "express";
import { createSchema, createYoga } from "graphql-yoga";
import { express as voyagerMiddleware } from "graphql-voyager/middleware/index.js";
import typeDefs from "./generated/schema.json" assert { type: "json" };

const server = express();
server.use("/voyager", voyagerMiddleware({ endpointUrl: "/graphql" }));

const yoga = createYoga({ schema: createSchema({ typeDefs }) });
server.use(yoga.graphqlEndpoint, yoga);

server.listen(4000, () => {
  console.log("GraphQL Server is listening on http://localhost:4000/graphql");
});
```

When I try to access http://localhost:4000/voyager I get "Unexpected reserved word at voyager:42:29" error in console.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=utf-8 />
  <meta name="viewport" content="user-scalable=no, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0">
  <title>GraphQL Voyager</title>
  <style>
    body {
      padding: 0;
      margin: 0;
```

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
No branches or pull requests

Notifications Customize

Unsubscribe

You're receiving notifications because you commented.

2 participants

GraphQL Weekly

What is GraphQL? Submit a link

A weekly newsletter of the best **news**, **articles** and **projects** about GraphQL

NAME Bob Loblaw EMAIL bob@example.com SUBSCRIBE

ISSUE 353 · MAY 17, 2024

ARTICLES

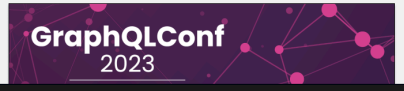
Build GraphQL-powered Generative AI applications with Amazon Bedrock and AWS AppSync

In this article, Ike Gabriel Yuson delves into the intersection of GraphQL and generative AI, demonstrating how Amazon Bedrock and AWS AppSync can be leveraged to build scalable and extensible applications. The author illustrates the benefits of using GraphQL's flexible schema and query capabilities to seamlessly integrate AI-generated content with existing data sources. By exploring real-world use cases and best practices, this article provides a comprehensive guide for developers seeking to harness the power of GraphQL in their generative AI projects.

SHARE THE GOODS

Found a cool GraphQL resource? Tell us all about it!

SUBMIT LINK



Apollo GraphQL

A community building flexible open source tools for GraphQL.

1.1k followers · San Francisco, CA · https://www.apollographql.com · @apollographql · https://discord.gg/yFZJH2QYrK · company/apollo-graphql · Verified

Follow

The Guild

A group of open source developers, bringing a different approach for sustainable open source. Our philosophy is to place our libraries under a person's name.

364 followers · http://the-guild.dev · @TheGuildDev · contact@the-guild.dev · Verified · Sponsor

Follow

UNIT TESTING

UNIT TESTING: REST

```
const express = require('express');
const app = express();
app.use(express.json());

let resources = [{ id: 1, name: 'Resource1' }];

app.get('/api/resources', (req, res) => {
  res.json(resources);
});

app.post('/api/resources', (req, res) => {
  const newResource = { id: resources.length + 1, name: req.bo
  resources.push(newResource);
  res.status(201).json(newResource);
```

UNIT TESTING: GRAPHQL

```
const { ApolloServer, gql } = require('apollo-server');

const typeDefs = gql`
  type Resource {
    id: ID!
    name: String!
  }

  type Query {
    resources: [Resource]
  }

  type Mutation {
    createResource(name: String!): Resource
```

UNIT TESTING: REST

```
const request = require('supertest');
const app = require('../app');
const { expect } = require('chai');

describe('REST API', () => {
  it('should get all resources', async () => {
    const res = await request(app).get('/api/resources');
    expect(res.status).to.equal(200);
    expect(res.body).to.be.an('array');
    expect(res.body.length).to.equal(1);
  });

  it('should create a new resource', async () => {
    const res = await request(app).post('/api/resources').send
```

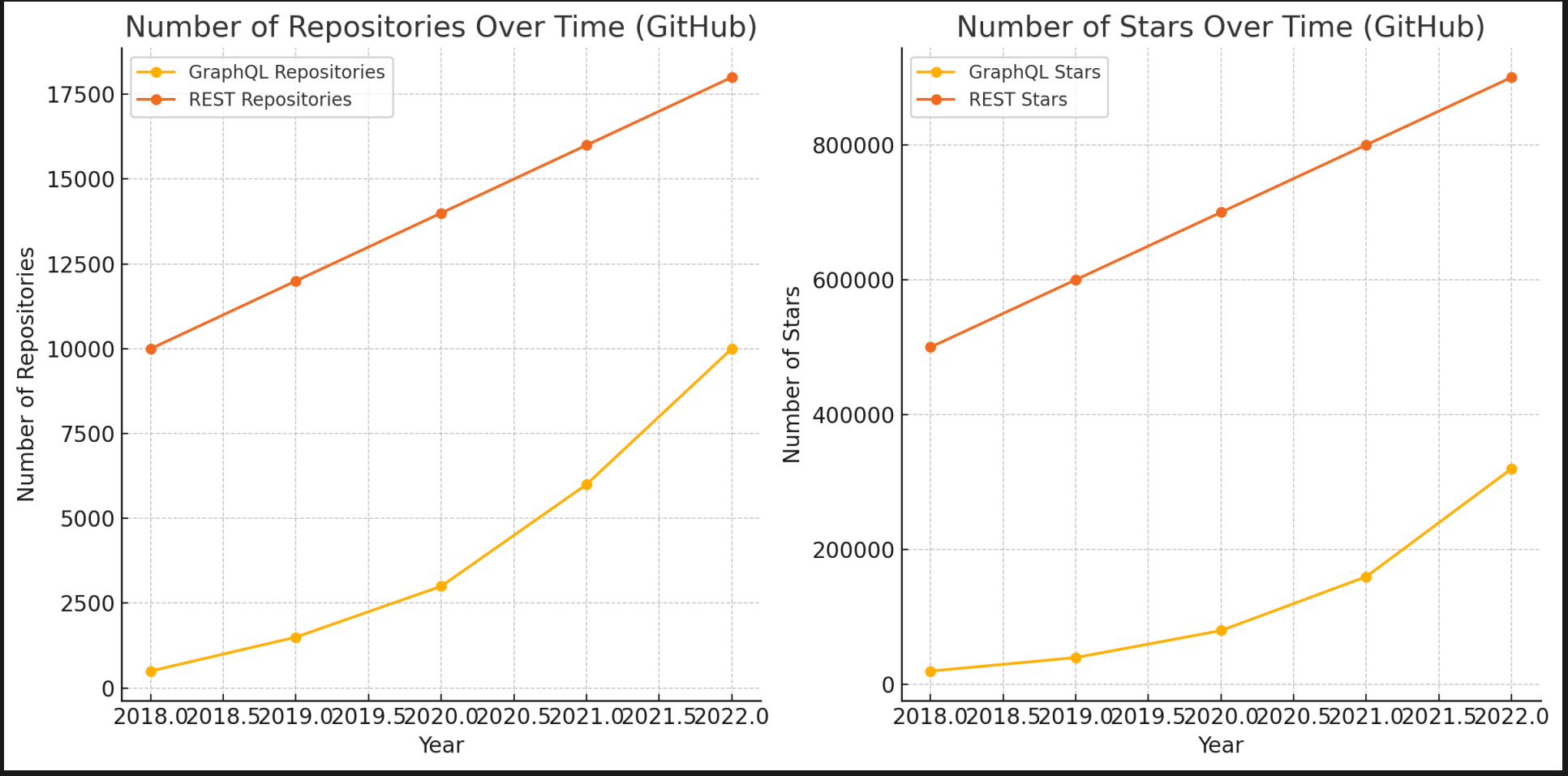
UNIT TESTING: GRAPHQL

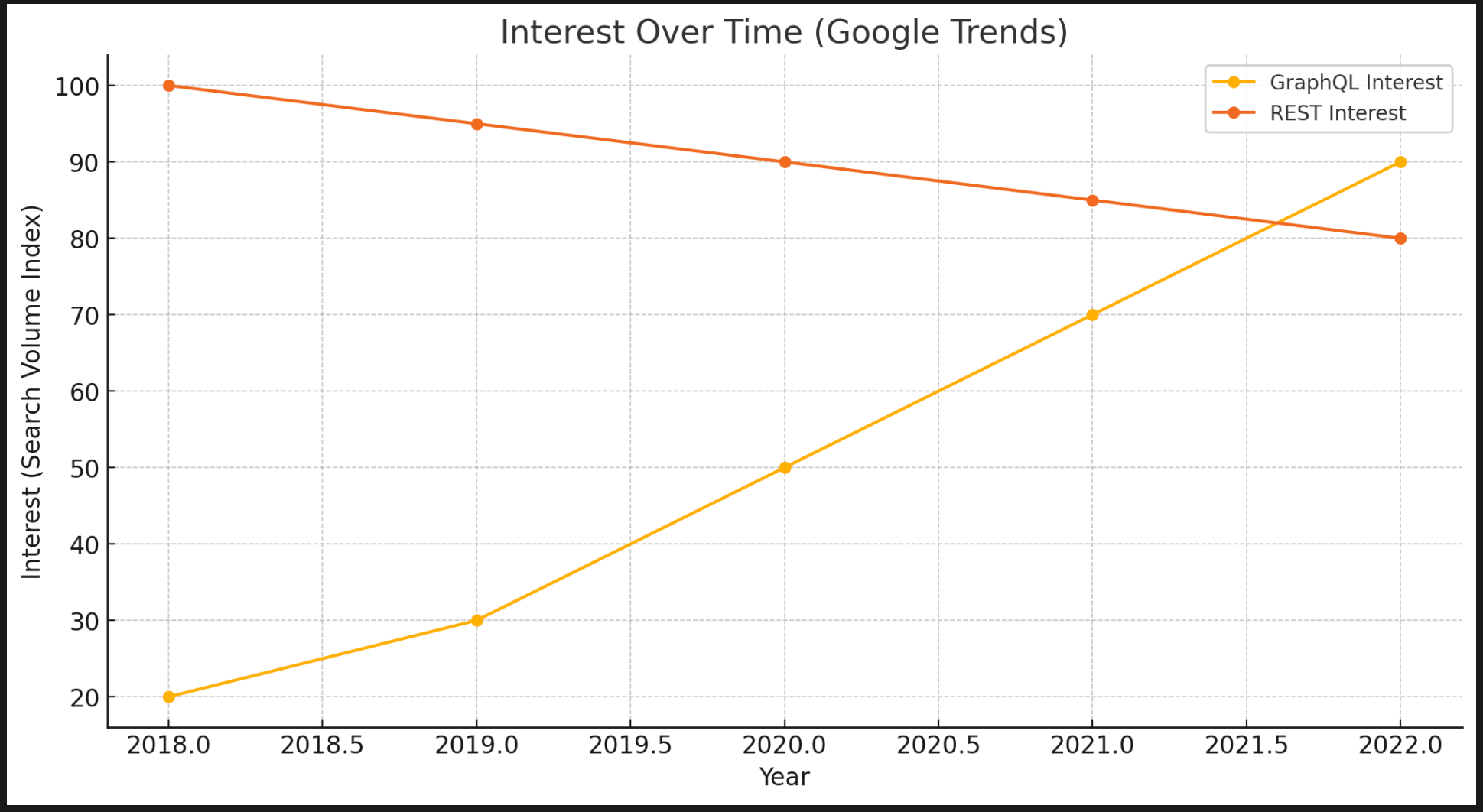
```
const { createTestClient } = require('apollo-server-testing');
const { expect } = require('chai');
const server = require('../schema');

const { query, mutate } = createTestClient(server);

describe('GraphQL API', () => {
  it('should get all resources', async () => {
    const GET_RESOURCES = `
      query {
        resources {
          id
          name
        }
```

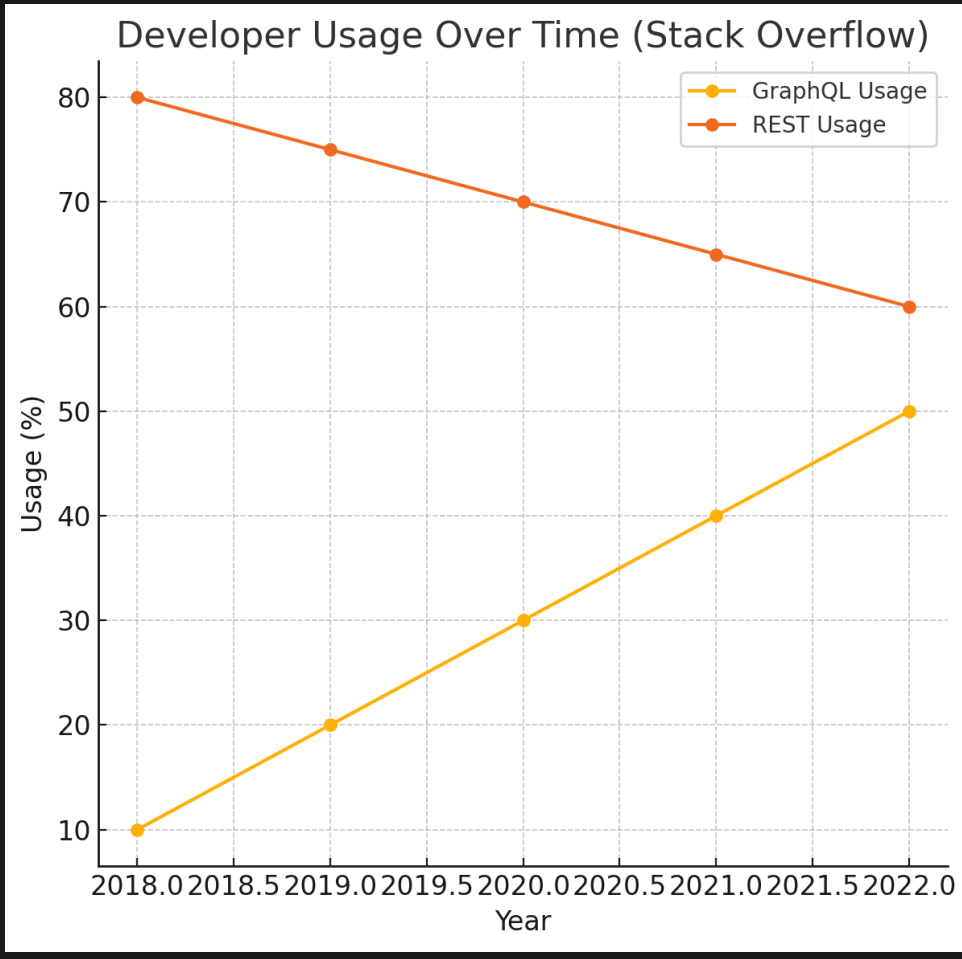
DEMO

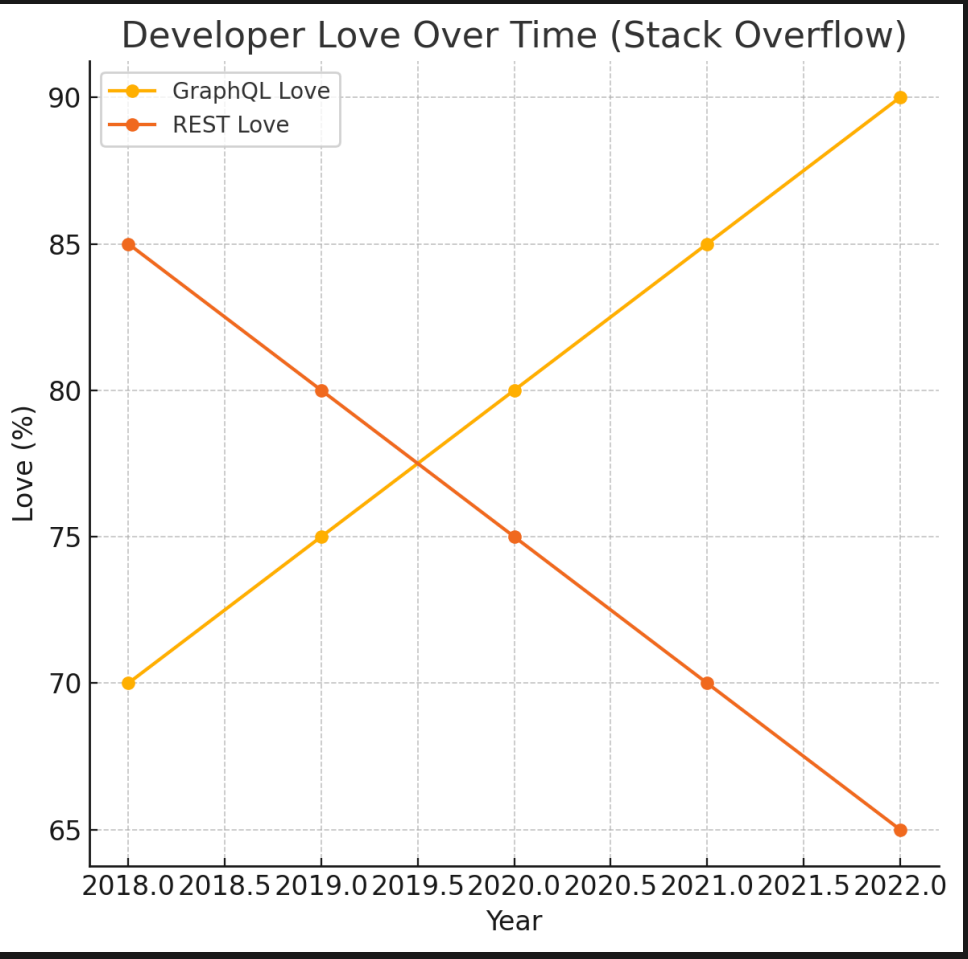
Unit Testing in GraphQL.

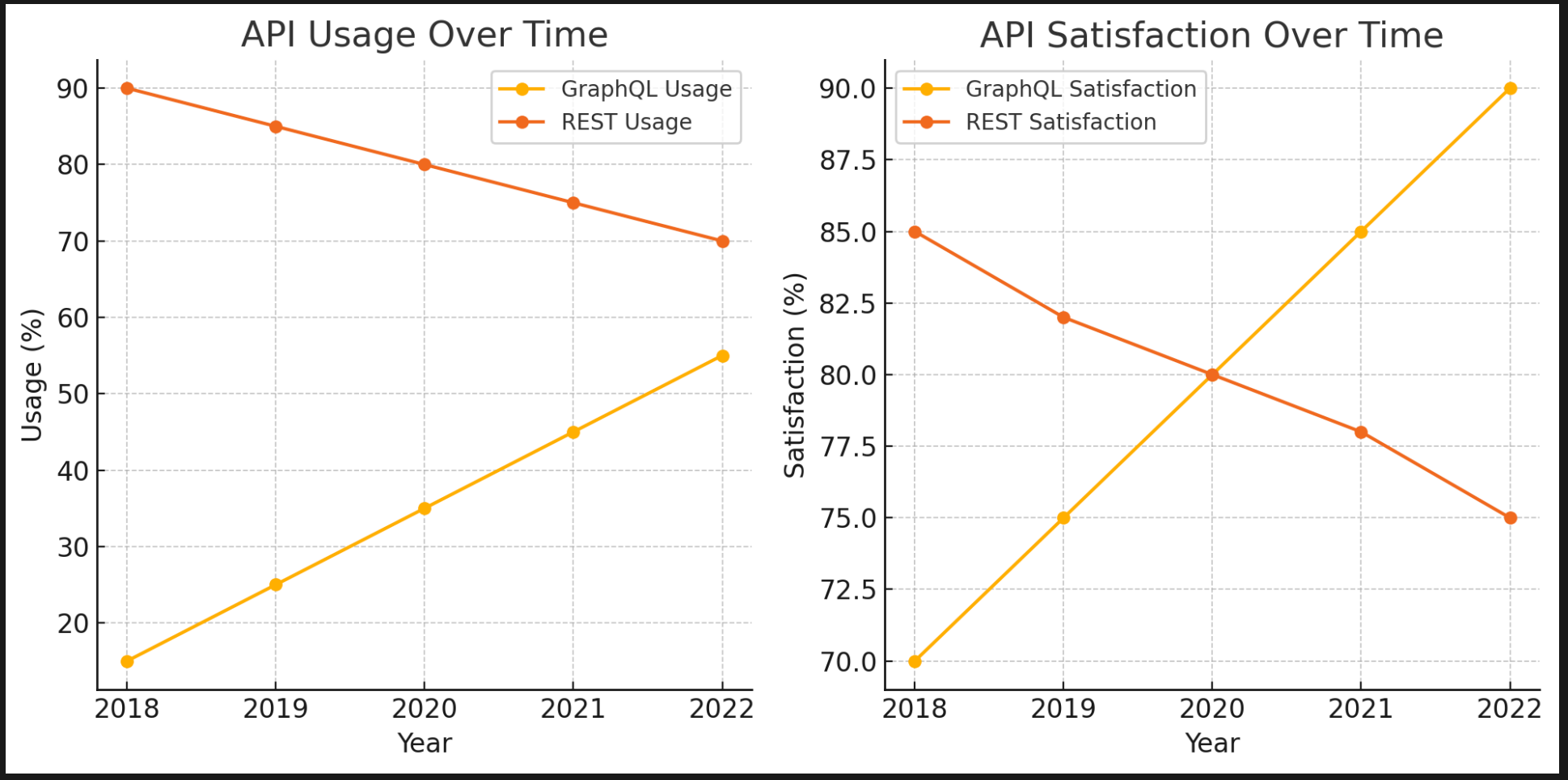
TRENDS


Number of Repositories Over Time (GitHub)
GraphQL Repositories
REST Repositories
Number of Repositories
0
2500
5000
7500
10000
12500
15000
17500
2018.0
2018.5
2019.0
2019.5
2020.0
2020.5
2021.0
2021.5
2022.0
Year
Number of Stars Over Time (GitHub)
GraphQL Stars
REST Stars
Number of Stars
0
200000
400000
600000
800000
2018.0
2018.5
2019.0
2019.5
2020.0
2020.5
2021.0
2021.5
2022.0
Year


Interest Over Time (Google Trends)
GraphQL Interest
REST Interest
Interest (Search Volume Index)
100
90
80
70
60
50
40
30
20
2018.0
2018.5
2019.0
2019.5
2020.0
2020.5
2021.0
2021.5
2022.0
Year


Developer Usage Over Time (Stack Overflow)
GraphQL Usage
REST Usage
Usage (%)
Year
10
20
30
40
50
60
70
80
2018.0
2018.5
2019.0
2019.5
2020.0
2020.5
2021.0
2021.5
2022.0


Developer Love Over Time (Stack Overflow)
GraphQL Love
REST Love
Love (%)
Year
65
70
75
80
85
90
2018.0
2018.5
2019.0
2019.5
2020.0
2020.5
2021.0
2021.5
2022.0


API Usage Over Time
GraphQL Usage
REST Usage
Usage (%)
90
80
70
60
50
40
30
20
2018
2019
2020
2021
2022
Year
API Satisfaction Over Time
GraphQL Satisfaction
REST Satisfaction
Satisfaction (%)
90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
70.0
2018
2019
2020
2021
2022
Year

CONCLUSION

USEFUL, BUT

- Caching
- Circular Queries (FE)
- Error Handling
- Introspection
- query *
- Up-to-date projects
- Unit Testing

Ditectrev

#Build Your Digital Future.

179 followers | https://education.ditectrev.com | https://shop.ditectrev.com | company/ditectrev | @ditectrev | https://discord.gg/RFjtXKfJy3 | contact@ditectrev.com

Unfollow

Pinned

Customize pins

Practice-Exams-Platform Public

Forked from eduardconstantin/azure-cloud-exams

Practice Exams (Web) Platform developed by Ditectrev's Community.

TypeScript ☆ 27 ⑂ 7

CloudMaster-Swift Public

CloudMaster Swift is an iOS application for studying cloud certifications developed by Ditectrev's Community.

Swift ☆ 6

Google-Cloud-Platform-GCP-Associate-Cloud-Engineer-Practice-Tests-Exams-Questions-Answers Public

⛳ PASS: Google Cloud Platform (GCP) Associate Cloud Engineer (ACE) by learning based on our Questions & Answers (Q&A) Practice Tests Exams.

☆ 68 ⑂ 36

Microsoft-Azure-AZ-500-Azure-Security-Engineer-Practice-Tests-Exams-Questions-Answers Public

⛳ PASS: Microsoft Azure AZ-500 (Azure Security Engineer Associate) by learning based on our Questions & Answers (Q&A) Practice Tests Exams.

☆ 4 ⑂ 12

ITIL-4-Foundation-IT-Service-Management-Practice-Tests-Exams-Questions-Answers Public

⛳ PASS: ITIL 4 Foundation (IT Service Management) by learning based on our Questions & Answers (Q&A) Practice Tests Exams.

☆ 23 ⑂ 10

Awesome-HTML-Book-HyperText-Markup-Language-HTML Public

Awesome HTML Book with practical CodeSandboxes.

☆ 5

View as: Public

You are viewing the README and pinned repositories as a public user.

You can create a README file visible to anyone.

Get started with tasks that most successful organizations complete.

Discussions

Set up discussions to engage with your community!

Turn on discussions

People

Invite someone

100% OpenSource: AWS, Azure, GCP, and more.

https://github.com/ditectrev